Project Report: Handwritten Digit Classification Using MNIST and Deep Learning

Objective

The objective of this project is to build a deep learning model using the MNIST dataset to classify handwritten digits (0–9). Additionally, it includes functionality to evaluate model performance and predict user-supplied digit images.

Packages Used and Their Purpose

`numpy`

- **Why used**: Efficient handling of numerical arrays and matrix operations.
- **How used**: Normalizing image pixel values and reshaping data.
- **Functionality solved**: Vectorized operations, crucial for preprocessing and prediction.

`matplotlib.pyplot` and `seaborn`

- **Why used**: For data visualization.
- **How used**:
    - Display misclassified digits.
    - Plot confusion matrix using seaborn's heatmap.
- **Functionality solved**: Makes model evaluation more interpretable via visualizations.

`tensorflow.keras`

- **Why used**: Provides a high-level API for building and training neural networks.
- **Modules used**:
    - `datasets.mnist`: Loads the handwritten digit dataset.
    - `models.Sequential`: Defines the sequential model architecture.

    - `layers`: Includes `Dense`, `Flatten`, and `Dropout` layers.
    - `utils.to_categorical`: Converts labels to one-hot encoded format.
- **Functionality solved**: Full pipeline from model building to training and evaluation.

`sklearn.metrics`

- **Why used**: For detailed performance metrics.
- **How used**:
    - `classification_report`: Shows precision, recall, and F1-score.
    - `confusion_matrix`: Evaluates model predictions against actual values.
- **Functionality solved**: Helps interpret model strengths and weaknesses across classes.

`cv2 (OpenCV)`

- **Why used**: For image reading, conversion, resizing, and preprocessing.
- **How used**:
    - Load user-uploaded image.
    - Convert to grayscale.
    - Resize to 28×28.
    - Normalize pixel values.
- **Functionality solved**: Prepares arbitrary user-supplied digit images for prediction.

Model Summary

- **Architecture**:
    - Input Layer: Flatten 28×28 image into a 784-element vector.

    - Hidden Layers: Dense layers with ReLU activations and dropout for regularization.
    - Output Layer: 10 neurons with softmax for digit classification.

- **Loss Function**: `categorical_crossentropy`
- **Optimizer**: `adam`
- **Metrics**: `accuracy`

Evaluation

- Achieved ~98% accuracy on test data.
- Confusion matrix and classification report indicate strong performance across all digit classes.

User Image Prediction

- Added functionality to allow external image input.
- Preprocesses uploaded digit images and predicts using the trained model.
- Visualization shows the predicted digit with its corresponding image.

Conclusion

This project demonstrates how deep learning can effectively classify handwritten digits using Keras. The integration of external image input enhances the practical utility of the model, making it adaptable for real-world handwriting recognition applications.